Self-Supervised State Representation for Reinforcement Learning-Assisted Evolutionary Algorithms

Xiaotong Liu, Ye Tian*, Hao Jiang, Langchun Si, Zimo Sheng, Xingyi Zhang
School of Computer Science and Technology, Anhui University, Hefei, China
*Corresponding author: field910921@gmail.com

***Abstract*—Reinforcement learning-assisted evolutionary algorithms are attracting increasing attention, but existing methods rely on handcrafted state representations that are often problem-dependent, limited in expressiveness, and lack robustness to variations in population composition and distribution. To address these limitations, a self-supervised state representation learning framework is proposed for reinforcement learning-assisted evolutionary algorithms. This framework follows a two-stage learning paradigm consisting of representation learning and policy learning, where the state encoder is trained in the representation learning stage and subsequently used to generate state inputs for reinforcement learning based policy inference. Specifically, a set-aware attention-based encoder is designed and trained via self-supervised contrastive learning using population objective values, while an auxiliary guided learning strategy with dynamic weighting is employed to stabilize the representation learning process. To evaluate the effectiveness of the proposed framework, it is integrated into representative reinforcement learning-assisted evolutionary algorithms for single- and multi-objective optimization problems to assess its generalization capability, with additional ablation studies analyzing key components. The results demonstrate the effectiveness of the proposed framework in addressing the limitations of handcrafted designs.**

I. Introduction

Reinforcement learning-assisted evolutionary algorithms have attracted increasing attention as a learning-to-optimize paradigm that enables adaptive control of evolutionary search [1]. In this paradigm, the evolutionary optimization process is typically modeled as a sequential decision-making problem [2], where reinforcement learning (RL) agents [3] interact with the evolutionary algorithm (EA) via key elements such as states, actions, and rewards. Specifically, the state characterizes the dynamic information of the evolving population, the action determines how algorithmic components and parameters are configured, and the reward evaluates the impact of the executed actions on the optimization process. While all these elements are important for the effectiveness of RL-assisted EAs, this work focuses on state representation [4], as it defines what the agent observes and serves as the input to both policy learning and value estimation, making an informative state essential for reliable action-value prediction and effective decision making during evolutionary search.

This work was supported in part by the National Natural Science Foundation of China (No. W2441019, No. 62522301, No. 62276001) and in part by the Anhui Provincial Natural Science Foundation (No. 2308085J03).

In RL-assisted EAs, state representation [5] is designed in different forms according to how population information is constructed and abstracted for decision making. Accordingly, existing state representations can be broadly categorized into three groups. The first category consists of handcrafted indicator-based state representations, where the population state is described using manually designed metrics, such as convergence, diversity, or feasibility-related indicators [6]. The second category adopts population-level statistical aggregation-based state representations, where state information is constructed by aggregating objective values of the population through predefined statistical functions, such as means, medians, standard deviations, or extrema [7]. The third category employs raw population-based state representations, where objective value or decision variable of the population are directly used as inputs to RL models, with little or no manual processing [8].

However, existing state representation methods still suffer from several fundamental limitations [9]. Handcrafted indicator-based state representations are highly dependent on prior knowledge and problem-specific assumptions, resulting in limited transferability across different optimization scenarios [10]. Population-level statistical aggregation-based representations reduce reliance on explicitly designed performance indicators, but inevitably compress comprehensive population information into a limited set of statistics, which restricts their ability to capture complex distributional structures [11]. Raw population-based state representations preserve more information by directly using objective value or decision variable matrices, yet they treat evolutionary populations as ordered tensors rather than unordered sets, making them sensitive to permutation, sampling variability, and stochastic perturbations and challenging for stable policy learning [12].

To address the limitations discussed above, we propose a self-supervised state representation learning framework (SSRL) for reinforcement learning-assisted evolutionary algorithms. The population objective values or decision variable matrices are input directly into the self-supervised learning module, which learns meaningful state representations without manual processing or predefined aggregation. In SSRL, attention mechanisms [13] are employed to ensure permutation-invariance by modeling evolutionary populations as unordered

sets, while contrastive learning [14] combined with augmentation strategies enhances robustness, allowing the model to effectively learn stable state representations despite variations in population dynamics, noise, and perturbations [15]. Additionally, guided information is incorporated to help the model focus on relevant features, while dynamic weighting adjusts the influence of the contrastive learning and guided signals throughout training, enabling SSRL to balance between robustness and targeted representation learning.

The main contributions of this work are as follows:

1) We propose a self-supervised state representation learning framework (SSRL) that addresses the limitations of existing methods in population state representation. By incorporating attention mechanisms, SSRL resolves the unordered set problem, ensuring permutation-invariance of evolutionary populations and enabling more flexible and stable state representations.
2) We introduce a multi-task learning approach that optimizes contrastive learning and guided information. Contrastive learning is applied with augmentation strategies to improve robustness by distinguishing positive and negative samples. Guided information provides additional supervision to focus on relevant population characteristics. Dynamic weighting balances the two tasks, adapting their importance during training to ensure effective learning of both robust features and task-specific representations.

The rest of this paper is organized as follows. Section II reviews state representation methods in RL-assisted EAs, including learning-based state representations. Section III presents the proposed SSRL framework. Section IV reports and analyzes the experimental results. Finally, Section V concludes the paper and discusses future work.

II. Related Work

Existing studies on state representation in RL-assisted EAs predominantly rely on handcrafted indicators to characterize the population search status. Xu *et al.* [16] proposed a Q-learning–based particle swarm optimization (QLPSO), in which handcrafted convergence and diversity indicators define a four-state representation for topology selection. Liu *et al.* [17] further proposed a three-stage RL framework, in which population states are manually defined using diversity, convergence, and fitness indicators, yielding an eight-state discrete representation for algorithm selection. In addition to indicator-based methods, some studies describe population states through statistical aggregation of population information. MOEA-ASF [7] represents population states by aggregating objective values of the population, including the maximum, minimum, median, and average values of each objective, which are directly used as inputs to the DQN. AGSEA [18] uses average sparsity, sparsity standard deviation, and sparsity distribution as inputs to a deep RL model for guiding vector selection. SMOEA-DQN [19] further incorporates individual sparsity dispersion and population-level sparsity variation for DQN-based operator selection.

To avoid information loss caused by manual abstraction or statistical aggregation, MOEA/D-DQN [8] directly uses raw decision variables of solutions together with their associated weight vectors as state inputs for deep RL, enabling neural networks to learn representations from high-dimensional population states. Similarly, RL-PSO [20] directly feeds raw particle position information, including current positions as well as personal and global best positions, into deep RL, allowing neural networks to capture complex swarm dynamics without explicit state abstraction. Overall, these state representation methods have inherent limitations. Indicator-based and aggregation-based approaches rely on prior knowledge and may discard useful structural information, while raw population-based representations often lead to high-dimensional and complex state spaces that challenge efficient RL. More importantly, these approaches generally do not explicitly address the representation learning problem itself. In contrast, deep learning–based [21] methods provide a natural mechanism for automatically extracting informative representations from complex population information, suggesting a promising direction for improving state design in RL-assisted EAs.

Beyond handcrafted state design, learning-based state representation methods have recently begun to attract attention in evolutionary optimization. In large-scale multi-objective optimization, Tian *et al.* [12] proposed an unsupervised neural network–based approach to learn low-dimensional representations of decision variables, where autoencoder-based models are employed to approximate the Pareto-optimal subspace and reduce the effective search space. Building upon this idea, Ming *et al.* [22] further introduced an unsupervised learning–based state representation method for RL-assisted EAs, in which a denoising autoencoder is used to learn population-level representations from objective vectors and provide informative states for deep RL. These studies indicate the feasibility of learning-based state representation in evolutionary optimization and RL-assisted EAs. However, most current methods adopt fixed unsupervised learning objectives (e.g., reconstruction-based autoencoders) [23] that are optimized independently of the downstream RL task. As a result, the learned representations may not be well aligned with decision-making requirements during the evolutionary process. Moreover, state representations are typically learned from limited forms of population information and trained in a static manner, which restricts their expressiveness and adaptability across different optimization stages or problem settings. These limitations suggest that state representation learning in RL-assisted EAs remains an open problem and can be further improved through more task-aware and adaptive learning mechanisms.

III. The Proposed Self-Supervised State Representation Learning

A. Self-Supervised Training of the State Encoder

Fig. 1 presents the overall pipeline for self-supervised training of the proposed state encoder, and the correspond-

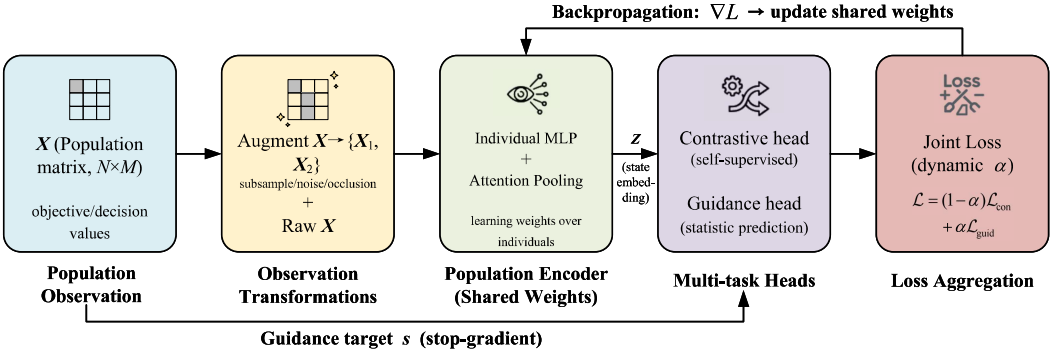


Fig. 1. Overall pipeline of the proposed SSRL framework. Population observations are transformed into multiple views and encoded by a shared attention-based population encoder. The encoder is trained via a joint self-supervised contrastive loss and an auxiliary guidance loss with dynamic weighting, producing compact and informative population-level state representations for downstream RL.

ing procedure is summarized in Algorithm 1. Starting from population observations, the framework learns a compact population-level state representation through an end-to-end training process. As illustrated in the figure, multiple views are first constructed from the same population observation and processed by a shared attention-based encoder to obtain latent state representations. These representations are then optimized using complementary self-supervised learning objectives, including contrastive learning across augmented views and guided information learning from the original observation. The resulting objectives are jointly optimized to train the state encoder, which can subsequently be used to generate stable and informative state representations without additional supervision.

Starting from a population observation, the state encoder is designed to transform an unordered set of individuals into a compact and permutation-invariant population-level representation. Let $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \in \mathbb{R}^{N \times M}$ denote the observation of a population consisting of N individuals with M-dimensional features. Each individual is first mapped into a latent embedding space by an individual embedding network $\phi(\cdot)$, yielding a set of embeddings $\mathbf{H} = \{\mathbf{h}_1, \ldots, \mathbf{h}_N\}$ with $\mathbf{h}_i \in \mathbb{R}^{d_h}$. Based on these embeddings, an attention scoring network $g(\cdot)$ assigns an importance score to each individual, which is normalized via a softmax operation to obtain attention weights:

$$a_i = \frac{\exp(g(\mathbf{h}_i))}{\sum_{j=1}^{N} \exp(g(\mathbf{h}_j))}. \tag{1}$$

Using the learned attention weights, the population-level representation is aggregated through a weighted pooling operation, i.e., $\mathbf{u} = \sum_{i=1}^{N} a_i \mathbf{h}_i$, which ensures permutation invariance with respect to the ordering of individuals. Finally, the aggregated representation $\mathbf{u}$ is projected into the state space by an output projection network $\psi(\cdot)$ to obtain the population state representation $\mathbf{z} = \psi(\mathbf{u})$. Here, $\psi(\cdot)$ denotes the output projection module of the state encoder, which maps the aggregated population representation into the latent state space. Moreover, Θ_{proj} denotes the parameters of the contrastive projection head, and $\rho_{\Theta_{\text{proj}}}(\cdot)$ denotes the corresponding contrastive projection head itself, which is used to map encoded states into the contrastive space for the computation of $\mathcal{L}_{\text{con}}$. Similarly, Θ_{guid} denotes the parameters of the guidance head, and $q_{\Theta_{\text{guid}}}(\cdot)$ denotes the corresponding guidance head, which is used to predict the guidance signal for the computation of $\mathcal{L}_{\text{guid}}$. Let Θ_{enc} denote the learnable parameters of the state encoder, including the individual embedding network $\phi(\cdot)$, the attention scoring network $g(\cdot)$, and the output projection network $\psi(\cdot)$.

Based on the encoded population state representations, the encoder is further optimized through a self-supervised learning scheme that combines contrastive learning and guided information learning. For contrastive learning, multiple augmented views are constructed from the same population observation using a set of stochastic augmentation strategies [24], including population permutation, feature masking, and noise perturbation [25]. These augmented observations are processed by the shared state encoder to obtain paired latent representations. Specifically, given two augmented views of the same population observation, their encoded representations are denoted by $\mathbf{z}_i^{(1)}$ and $\mathbf{z}_i^{(2)}$, respectively. They are then mapped by the contrastive projection head to obtain $\mathbf{p}_i^{(1)} = \rho_{\Theta_{\text{proj}}}(\mathbf{z}_i^{(1)})$ and $\mathbf{p}_i^{(2)} = \rho_{\Theta_{\text{proj}}}(\mathbf{z}_i^{(2)})$. The contrastive loss $\mathcal{L}_{\text{con}}$ is computed using a symmetric InfoNCE objective [26], where the two projected representations derived from the same population observation form a positive pair, while projected representations from different population observations in the mini-batch are treated as negative pairs. Cosine similarity is adopted as the similarity measure, and τ denotes the temperature parameter.

Meanwhile, guided information learning is introduced to inject informative population-level priors into the representation learning process. A guidance signal $\mathbf{s}_i$ is extracted from the original population observation $\mathbf{X}_i$ by a predefined guidance signal extractor $G(\cdot)$ and is treated as a fixed target during training. It is worth noting that the guidance signal is not restricted to a specific formulation and can be flexibly designed according to different optimization scenarios. The encoded state representation $\mathbf{z}_i$ is mapped to a prediction $\hat{\mathbf{s}}_i = q_{\Theta_{\text{guid}}}(\mathbf{z}_i)$ through the guidance head, and the discrepancy between $\hat{\mathbf{s}}_i$ and

Algorithm 1 SSRL

Input: $\mathcal{B}$ (mini-batches of population observations), $\mathcal{T}$ (observation transformations), $G(\cdot)$ (guidance signal extractor), $\alpha(\cdot)$ (dynamic weight schedule), η (learning rate), τ (temperature parameter)
Output: Θ^*_{enc}

Initialize: $\Theta_{\text{enc}}, \Theta_{\text{proj}}, \Theta_{\text{guid}}$; iter $\leftarrow 0$
while not converged **do**
  // Sample a mini-batch of population observations
  $\{\mathbf{X}_i\}_{i=1}^{B} \leftarrow \texttt{SampleBatch}(\mathcal{B})$
  // Generate two augmented views and keep the raw observation
  for $i = 1, \ldots, B$ **do**
    $(\mathbf{X}_i^{(1)}, \mathbf{X}_i^{(2)}) \leftarrow \texttt{Augment}(\mathbf{X}_i; \mathcal{T})$
    $\mathbf{s}_i \leftarrow G(\mathbf{X}_i)$
  end for
  // Shared population encoder forward pass
  for $i = 1, \ldots, B$ **do**
    $\mathbf{z}_i^{(1)} \leftarrow \texttt{Encoder}(\mathbf{X}_i^{(1)}; \Theta_{\text{enc}})$
    $\mathbf{z}_i^{(2)} \leftarrow \texttt{Encoder}(\mathbf{X}_i^{(2)}; \Theta_{\text{enc}})$
    $\mathbf{z}_i \leftarrow \texttt{Encoder}(\mathbf{X}_i; \Theta_{\text{enc}})$
  end for
  // Compute joint loss (contrastive + guided, details in Algorithm 2)
  $\alpha_{\text{iter}} \leftarrow \alpha(\text{iter})$
  $\mathcal{L} \leftarrow \texttt{ComputeJointLoss}($
    $\{\mathbf{z}_i, \mathbf{z}_i^{(1)}, \mathbf{z}_i^{(2)}, \mathbf{s}_i\}_{i=1}^{B}, \alpha_{\text{iter}}, \Theta_{\text{proj}}, \Theta_{\text{guid}}, \tau)$
  // Backpropagation: update shared weights
  $\Theta_{\text{enc}}, \Theta_{\text{proj}}, \Theta_{\text{guid}} \leftarrow \texttt{Update}(\Theta_{\text{enc}},$
    $\Theta_{\text{proj}}, \Theta_{\text{guid}}, \nabla\mathcal{L}, \eta)$
  iter $\leftarrow$ iter $+ 1$
end while
return $\Theta^*_{\text{enc}} \leftarrow \Theta_{\text{enc}}$

$\mathbf{s}_i$ is minimized by a regression objective,

$$\mathcal{L}_{\text{guid}} = \frac{1}{B}\sum_{i=1}^{B} \|\hat{\mathbf{s}}_i - \mathbf{s}_i\|_2^2. \tag{2}$$

By leveraging such auxiliary supervision, the encoder is encouraged to preserve informative population-level characteristics that are potentially relevant to downstream decision making, without relying on explicit labels.

The contrastive learning objective and the guided information learning objective are jointly optimized through a unified training objective. Specifically, the overall loss is defined as

$$\mathcal{L} = (1 - \alpha_t)\mathcal{L}_{\text{con}} + \alpha_t \mathcal{L}_{\text{guid}}, \tag{3}$$

where α is a dynamic weighting coefficient that controls the contribution of the guidance signal during training. In the early stage, a relatively large α is adopted to emphasize guided information learning, which helps stabilize representation learning and provides informative inductive bias. As training progresses, α gradually decreases, allowing the encoder to rely more on contrastive learning to autonomously capture

Algorithm 2 Joint contrastive–guided loss computation

Input: $\{\mathbf{z}_i, \mathbf{z}_i^{(1)}, \mathbf{z}_i^{(2)}, \mathbf{s}_i\}_{i=1}^{B}$, α_{iter}, Θ_{proj}, Θ_{guid}, τ
Output: $\mathcal{L}$

// Contrastive projections for two augmented views
for $i = 1, \ldots, B$ **do**
  $\mathbf{p}_i^{(1)} \leftarrow \rho_{\Theta_{\text{proj}}}(\mathbf{z}_i^{(1)})$
  $\mathbf{p}_i^{(2)} \leftarrow \rho_{\Theta_{\text{proj}}}(\mathbf{z}_i^{(2)})$
end for
// InfoNCE contrastive loss with in-batch negatives
$\mathcal{L}_{\text{con}} \leftarrow \texttt{InfoNCE}(\{\mathbf{p}_i^{(1)}, \mathbf{p}_i^{(2)}\}_{i=1}^{B}, \tau)$
// Guided prediction loss
for $i = 1, \ldots, B$ **do**
  $\hat{\mathbf{s}}_i \leftarrow q_{\Theta_{\text{guid}}}(\mathbf{z}_i)$
end for
$\mathcal{L}_{\text{guid}} \leftarrow \frac{1}{B}\sum_{i=1}^{B} \|\hat{\mathbf{s}}_i - \mathbf{s}_i\|_2^2$
// Adaptive loss combination
$\mathcal{L} = (1 - \alpha_{\text{iter}})\mathcal{L}_{\text{con}} + \alpha_{\text{iter}}\mathcal{L}_{\text{guid}}$
return $\mathcal{L}$

intrinsic population-level structures. α is scheduled using a cosine decay strategy,

$$\alpha_t = \frac{1}{2}\left(1 + \cos\left(\pi \frac{t}{T}\right)\right), \tag{4}$$

where t denotes the current training iteration and T controls the decay horizon. The joint loss is optimized in an end-to-end manner, as summarized in Algorithm 2.

B. Integration of the State Encoder into RL-Assisted EA

As illustrated in Fig. 2, the integration of the trained state encoder into the RL-assisted EA is organized into two stages: representation learning and policy learning. In the representation learning stage, the EA is executed with randomly selected actions, corresponding to different component or parameter configurations. This stage is solely devoted to collecting population-level observations, which are stored in an observation buffer and used as training data for the self-supervised state encoder. Notably, the encoder is trained exclusively on population-level observations and does not require action, reward, or state-transition information from reinforcement learning.

After representation learning, the trained state encoder is used to map population observations into RL states, enabling both experience replay initialization and subsequent policy learning. During the policy learning stage, the state encoder is kept fixed and serves as a deterministic state construction module. Therefore, the additional overhead of SSRL mainly comes from offline observation collection and encoder training, together with online state encoding. Since SSRL is instantiated in different RL-assisted EAs, the overall time complexity depends on the specific downstream algorithm, while the extra memory overhead mainly comes from the encoder parameters and training mini-batches. As the RL-assisted EA proceeds, newly generated population observations are continuously

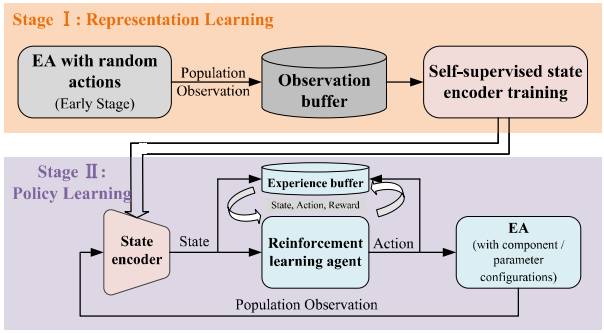


Fig. 2. Two-stage integration of the proposed SSRL framework into RL-assisted EAs. The population state encoder is first trained via self-supervised learning from population observations, and then incorporated into the policy learning stage to provide informative state representations for RL–based adaptive configuration.

encoded into states, while the policy is updated following standard RL-assisted evolutionary optimization procedures.

IV. Experiments Studies

To verify the performance of the proposed SSRL, it is applied to four representative RL-assisted EAs covering different optimization scenarios, including single-objective optimization (QLPSO [16] and TSRL-CMM [17]), and multi-objective optimization (MOEA/D-DQN [8] and MOEA-ASF [7]). TSRL-CMM employs Q-learning to adaptively select one of four metaheuristic algorithms based on the population state, while QLPSO also adopts Q-learning to select one of four neighborhood topologies in particle swarm optimization. MOEA/D-DQN and MOEA-ASF both adopt DQN-based decision-making, where the former selects evolutionary operators and the latter selects environmental selection strategies. Based on the above methods, the proposed SSRL is embedded into existing RL-assisted EAs by replacing their original state representations. For each downstream algorithm, SSRL is trained separately from scratch using population observations collected from the corresponding algorithmic process, rather than sharing a single pretrained encoder across different algorithms. The state encoder is trained from population-level observations, where population objective values are directly used as inputs for both single-objective and multi-objective optimization. During training, statistical descriptors of the input, including the mean, standard deviation, decile-based distribution, and entropy, are adopted as guidance information.

A. Settings of Algorithms

For fair comparisons, all baseline algorithms and their SSRL-embedded versions follow the same parameter settings as reported in their original papers. For TSRL-CMM, the population size is set to $N = 50$ and the discount factor is $\gamma = 0.8$. For QLPSO, the population size is $N = 30$, with $w = 0.729844$ and $c = 1.49618$, where the learning rate α decreases linearly from 1 to 0.1, and $\gamma = 0.8$. For MOEA/D-DQN, the population size is set to $N = 105$ and the neighborhood size for the Tchebycheff approach is set to 20. For MOEA-ASF, the population size is $N = 100$, and the parameter λ is set to 30% of the total number of generations. The parameter settings of the proposed SSRL framework are summarized in Table I.

TABLE I
Parameter Settings of the Proposed SSRL

Parameter	Value
Encoder input	Population-level observation matrix
Encoder hidden layers	2 layers (64 units, ReLU)
Pooling layer	Attention-based pooling
State embedding layer	1 layer (d_{enc} units, ReLU)
State embedding dimension	$d_{\text{enc}} = 3 * M$
Projection head	1 layer (64 units, ReLU)
Guide prediction head	1 layer (13 units, linear)
Guide output	13-dimensional vector
Optimizer	Adam
Learning rate	1.0×10^{-3}
Batch size	32
Training epochs	30
Contrastive temperature τ	0.2
Random exploration phase	$0.2 \times (\text{FEs}/\text{MaxFEs})$

TABLE II
Results of Single-Objective Optimization Problems (F1–F10).

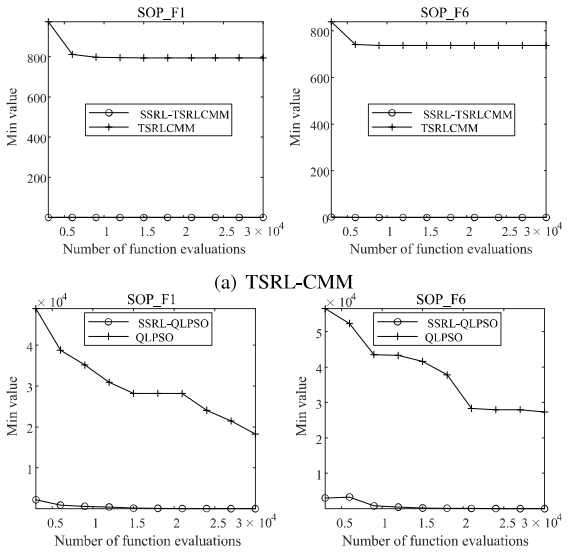
Problem	D	QLPSO		TSRL-CMM	
		Base	SSRL	Base	SSRL
SOP_F1	30	2.2589e+2 (1.88e+2) −	1.2894e-18 (6.27e-18)	5.3011e-2 (1.31e-1) −	9.8077e-31 (5.37e-31)
SOP_F2	30	4.6216e+0 (2.74e+0) −	1.6422e-11 (5.77e-11)	8.3745e-2 (1.00e-1) −	1.4338e-22 (3.79e-21)
SOP_F3	30	4.4881e+3 (1.86e+3) −	3.6885e-4 (1.22e-3)	4.5682e+0 (6.95e+0) −	3.0278e-32 (1.05e-31)
SOP_F4	30	1.7674e+1 (5.75e+0) −	1.2259e-7 (3.73e-7)	1.2070e+0 (8.43e-1) −	1.9991e-21 (7.41e-21)
SOP_F5	30	2.9557e+4 (2.62e+4) −	5.7402e+0 (3.33e-1)	6.4089e+1 (1.67e+2) −	4.4623e-5 (1.02e-4)
SOP_F6	30	2.0150e+2 (1.43e+2) −	0.0000e+0 (0.00e+0)	3.5000e+0 (3.54e+0) −	0.0000e+0 (0.00e+0)
SOP_F7	30	3.4571e-1 (2.03e-1) −	2.5116e-2 (1.72e-2)	2.0864e-2 (1.21e-2) ≈	2.0842e-2 (1.30e-2)
SOP_F8	30	-2.0704e+3 (4.51e+2) −	-2.7225e+3 (8.25e+2)	-2.5453e+3 (3.52e+2) −	-4.1898e+3 (2.19e-6)
SOP_F9	30	8.5540e+1 (1.63e+1) −	3.7800e+1 (1.67e+1)	9.4533e+0 (4.61e+0) −	0.0000e+0 (0.00e+0)
SOP_F10	30	7.3613e+0 (2.02e+0) −	3.7020e-12 (7.77e-12)	2.0907e+0 (9.85e-1) −	4.4409e-16 (0.00e+0)
+/ − / ≈		0/10/0		0/9/1	

B. Settings of Problems

For single-objective optimization, a set of 23 fixed-dimensional benchmark functions [27] is adopted, among which the first ten functions (F1–F10) are selected for evaluation, with the maximum number of function evaluations set to 30 000. For multi-objective optimization, the WFG benchmark problems [28] are employed, where the number of objectives is set to 3, and the number of decision variables is set to 12 for WFG1–WFG9, with 30 000 function evaluations. All experiments are implemented in MATLAB and executed on the evolutionary multi-objective optimization platform PlatEMO v4.13 [29]. Each algorithm is executed independent runs on each test instance. The quality of the obtained solutions is assessed using the inverted generational distance (IGD) metric [30], where 10 000 reference points are uniformly sampled from each Pareto front to calculate the IGD value. In addition, the Wilcoxon rank-sum test [31] with a significance level of 0.05 is performed to analyze the statistical differences between the results. In the comparison, "+", "–", and "≈" indicate that the result of a compared baseline algorithm is significantly better, significantly worse, or statistically similar to that of the SSRL-embedded versions, respectively.

TABLE III
RESULTS OF MULTI-OBJECTIVE OPTIMIZATION PROBLEMS (WFG1–WFG9).

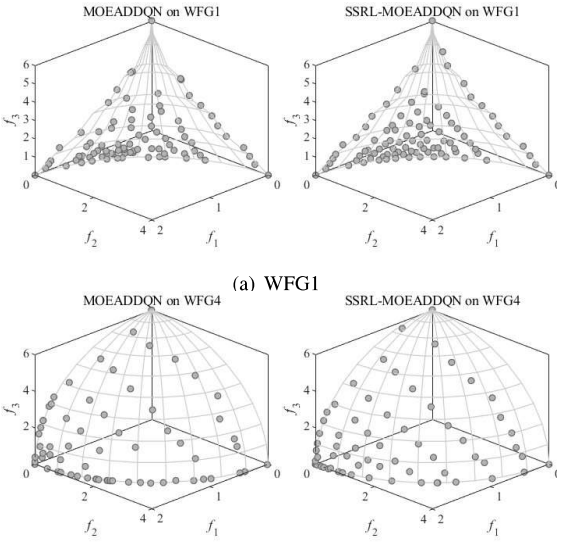
Problem	M	D	MOEA/D-DQN		MOEA-ASF	
			Base	SSRL	Base	SSRL
WFG1	3	12	3.5346e-1 (1.81e-1) −	1.9091e-1 (8.98e-3)	1.5727e+0 (4.99e-2) −	6.0193e-1 (1.51e-1)
WFG2	3	12	2.9577e-1 (3.01e-2) −	2.6590e-1 (1.06e-2)	2.1642e-1 (1.27e-2) +	2.9846e-1 (4.26e-2)
WFG3	3	12	1.4002e-1 (5.31e-3) −	1.3530e-1 (1.07e-3)	2.7182e-1 (3.68e-2) −	1.4043e-1 (1.64e-2)
WFG4	3	12	3.4615e-1 (2.58e-2) −	3.1653e-1 (1.25e-2)	3.5349e-1 (1.86e-2) −	3.1251e-1 (1.27e-2)
WFG5	3	12	3.0682e-1 (8.83e-3) −	2.9108e-1 (1.78e-2)	2.7985e-1 (2.87e-2) +	3.1372e-1 (6.14e-3)
WFG6	3	12	3.3236e-1 (2.62e-2) −	3.1308e-1 (2.89e-2)	3.7422e-1 (4.11e-2) −	2.8620e-1 (2.59e-2)
WFG7	3	12	3.0994e-1 (1.36e-2) −	2.9089e-1 (1.59e-2)	3.3599e-1 (2.63e-2) −	2.8702e-1 (8.50e-3)
WFG8	3	12	3.6954e-1 (9.65e-3) −	3.6049e-1 (8.03e-3)	4.3289e-1 (2.10e-2) −	3.7312e-1 (8.26e-3)
WFG9	3	12	3.5612e-1 (5.23e-2) ≈	3.5213e-1 (5.03e-2)	3.6538e-1 (5.16e-2) −	3.3753e-1 (5.71e-2)
+/ − / ≈			0/8/1		2/7/0	



(a) TSRL-CMM

(b) QLPSO

Fig. 3. Convergence curves of TSRL-CMM and QLPSO with and without SSRL on single-objective optimization problems



(a) WFG1

(b) WFG4

Fig. 4. Final solution sets of MOEA/D-DQN and SSRL-MOEA/D-DQN corresponding to the median IGD value over 30 independent runs.

TABLE IV
ABLATION STUDY OF THE PROPOSED SSRL FRAMEWORK ON MOEA/D-DQN OVER WFG1–WFG9.

Method	WFG1–3	WFG4–6	WFG6–9
MOEA/D-DQN (Full SSRL)	1.3538e-01	2.2451e-01	1.3538e-01
no Attention	2.2817e+01	1.0434e+00	2.2866e+00
no Contrastive Learning	1.5516e+00	2.5346e-01	2.4002e+00
no Guidance Learning	4.9091e+01	2.5311e+01	2.2180e+00

C. Results on Benchmark Problems

The experimental results demonstrate that embedding the proposed SSRL framework into six representative RL-assisted EAs leads to consistent performance improvements across diverse optimization scenarios. As shown in Table II, the most significant improvements are observed on single-objective optimization problems, where SSRL consistently enhances both QLPSO and TSRL-CMM, indicating that automatically learned population-level state representations are particularly effective for guiding decision-making in relatively simple optimization landscapes. As reported in Tables III, for multi-objective optimization, the SSRL-embedded algorithms generally achieve performance comparable to their baseline counterparts, suggesting that although overall performance is strongly influenced by the choice of actions (e.g., evolutionary operators or environmental selection strategies), the proposed SSRL is capable of providing state representations that are sufficiently informative to reach the performance level of carefully designed manual states.

In addition, as illustrated in Fig. 3, the convergence curves of TSRL-CMM and QLPSO on single-objective optimization problems show that their SSRL-enhanced variants achieve faster and better convergence toward optimal solutions than the original algorithms. As shown in Fig. 4, the solution sets obtained by SSRL-MOEA/D-DQN exhibit a comparable and well-distributed approximation of the Pareto front on WFG1 and WFG4, indicating that the proposed SSRL framework preserves the original search behavior and solution distribution of MOEA/D-DQN.

D. Ablation Experiments

Ablation studies are conducted by embedding different variants of the proposed SSRL framework into MOEA/D-DQN to investigate the contribution of each key component. Specifically, three variants are considered by removing the attention mechanism, contrastive learning objective, and guidance learning module, respectively. To provide a clearer comparison among different ablation variants, the results on WFG1–WFG9 are further summarized by averaging the IGD values on WFG1–WFG3, WFG4–WFG6 and WFG7–WFG9.

As reported in Table IV, the full SSRL framework consistently achieves the best performance across both problem groups, while removing any individual component leads to noticeable performance degradation. In particular, the removal

of the guidance learning module results in the most severe degradation, indicating its critical role in stabilizing and guiding the learning of population-level state representations. These results confirm that the proposed SSRL framework benefits from the joint effect of its components, and that the integrated design effectively supports policy learning in MOEA/D-DQN.

V. Conclusions

To address the limitations of manually designed state representations in RL-assisted EAs, this paper proposed a self-supervised state representation learning (SSRL) framework that learns population-level representations directly from raw population observations. By combining attention-based pooling, contrastive learning, and guidance learning, SSRL can capture permutation-invariant population information and incorporate useful guidance signals during training. The framework was embedded into four representative RL-assisted EAs for single-objective and multi-objective optimization. Experimental results show that SSRL consistently improves or maintains competitive performance over the original algorithms, and ablation studies further verify the effectiveness of the proposed design.

In future work, several directions can be further explored to extend the applicability of the proposed framework. One promising direction is to apply SSRL to a wider range of optimization scenarios, including more complex problem settings and real-world engineering applications [32], [33]. Another important direction is to investigate the integration of SSRL with multi-agent RL, where more advanced state representation mechanisms may be required to support coordinated decision-making [34]. These directions may further enhance the generality and effectiveness of learning-based state representations for EAs.

References

[1] Y. Song, Y. Wu, Y. Guo, et al., "Reinforcement learning-assisted evolutionary algorithm: A survey and research opportunities," Swarm Evol. Comput., vol. 86, p. 101517, 2024.

[2] K. Tang and X. Yao, "Learn to optimize—a brief overview," Natl. Sci. Rev., vol. 11, no. 8, p. nwae132, 2024.

[3] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," J. Artif. Intell. Res., vol. 4, pp. 237–285, 1996.

[4] T. de Bruin, J. Kober, K. Tuyls, and R. Babuska, "Integrating state representation learning into deep reinforcement learning," IEEE Robot. Autom. Lett., vol. 3, no. 3, pp. 1394–1401, 2018.

[5] A. Anand, E. Racah, S. Ozair, Y. Bengio, and A. Courville, "Unsupervised state representation learning in atari," in Proc. Adv. Neural Inf. Process. Syst., vol. 32, 2019.

[6] S. Shao, Y. Tian, Y. Zhang, et al., "Knowledge learning-based dimensionality reduction for solving large-scale sparse multiobjective optimization problems," IEEE Trans. Cybern., in press.

[7] Y. Tian, L. Yao, S. Shao, et al., "Deep reinforcement learning based adaptive environmental selection for evolutionary multi-objective optimization," in Proc. IEEE Congr. Evol. Comput. (CEC), 2024, pp. 1–8.

[8] Y. Tian, X. Li, H. Ma, et al., "Deep reinforcement learning based adaptive operator selection for evolutionary multi-objective optimization," IEEE Trans. Emerg. Top. Comput. Intell., vol. 7, no. 4, pp. 1051–1064, 2022.

[9] N. W. Schuck, R. Wilson, and Y. Niv, "A state representation for reinforcement learning and decision-making in the orbitofrontal cortex," in *Goal-Directed Decision Making*. Cambridge, MA, USA: Academic Press, 2018, pp. 259–278.

[10] N. Botteghi, M. Poel, and C. Brune, "Unsupervised representation learning in deep reinforcement learning: A review," IEEE Control Syst., vol. 45, no. 2, pp. 26–68, 2025.

[11] F. Liu, R. Tang, X. Li, et al., "State representation modeling for deep reinforcement learning based recommendation," Knowl.-Based Syst., vol. 205, p. 106170, 2020.

[12] Y. Tian, L. Wang, S. Yang, et al., "Neural network-based dimensionality reduction for large-scale binary optimization with millions of variables," IEEE Trans. Evol. Comput., 2024.

[13] Z. Niu, G. Zhong, and H. Yu, "A review on the attention mechanism of deep learning," Neurocomputing, vol. 452, pp. 48–62, 2021.

[14] P. Khosla, P. Teterwak, C. Wang, et al., "Supervised contrastive learning," in Proc. Adv. Neural Inf. Process. Syst., vol. 33, pp. 18661–18673, 2020.

[15] X. Wang and G. J. Qi, "Contrastive learning with stronger augmentations," IEEE Trans. Pattern Anal. Mach. Intell., vol. 45, no. 5, pp. 5549–5560, 2022.

[16] Y. Xu and D. Pi, "A reinforcement learning-based communication topology in particle swarm optimization," Neural Comput. Appl., vol. 32, no. 14, pp. 10007–10032, 2020.

[17] X. Liu, T. Wang, Z. Zeng, et al., "Three stage based reinforcement learning for combining multiple metaheuristic algorithms," Swarm Evol. Comput., vol. 95, p. 101935, 2025.

[18] S. Shao, Y. Tian, and X. Zhang, "Deep reinforcement learning assisted automated guiding vector selection for large-scale sparse multi-objective optimization," Swarm Evol. Comput., vol. 88, p. 101606, 2024.

[19] P. Zhang, L. Wang, J. Rong, et al., "An evolutionary learning assisted autonomous selection of sparsity-aware genetic operators for sparse large-scale multi-objective optimization," Tsinghua Sci. Technol., vol. 31, no. 1, pp. 379–398, 2026.

[20] Wu D, Wang G G. Employing reinforcement learning to enhance particle swarm optimization methods[J]. Engineering Optimization, 2022, 54(2): 329-348.

[21] J. Pei, Y. Mei, J. Liu, et al., "Adaptive operator selection for metaheuristics: A survey," IEEE Trans. Artif. Intell., 2025.

[22] F. Ming, W. Gong, B. Xue, et al., "Automated configuration of evolutionary algorithms via deep reinforcement learning for constrained multiobjective optimization," IEEE Trans. Cybern., in press.

[23] Z. Wang, C. Wang, C. Gao, et al., "An evolutionary autoencoder for dynamic community detection," Sci. China Inf. Sci., vol. 63, no. 11, p. 212205, 2020.

[24] E. D. Cubuk, B. Zoph, D. Mane, et al., "Autoaugment: Learning augmentation strategies from data," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), 2019, pp. 113–123.

[25] E. Nardone, T. D'Alessandro, C. De Stefano, et al., "How data augmentation affects evolutionary algorithms in feature selection: An experimental study," SN Comput. Sci., vol. 6, no. 5, p. 536, 2025.

[26] T. Chen, S. Kornblith, M. Norouzi, et al., "A simple framework for contrastive learning of visual representations," in Proc. 37th Int. Conf. Mach. Learn. (ICML), 2020, pp. 1597–1607.

[27] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," IEEE Trans. Evol. Comput., vol. 3, no. 2, pp. 82–102, 1999.

[28] S. Huband, P. Hingston, L. Barone, and L. While, "A review of multiobjective test problems and a scalable test problem toolkit," IEEE Trans. Evol. Comput., vol. 10, no. 5, pp. 477–506, 2006.

[29] Y. Tian, W. Zhu, X. Zhang, and Y. Jin, "A practical tutorial on solving optimization problems via PlatEMO," Neurocomputing, vol. 518, pp. 190–205, 2023.

[30] E. Zitzler, L. Thiele, M. Laumanns, et al., "Performance assessment of multiobjective optimizers: An analysis and review," IEEE Trans. Evol. Comput., vol. 7, no. 2, pp. 117–132, 2003.

[31] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," Swarm Evol. Comput., vol. 1, no. 1, pp. 3–18, 2011.

[32] S. Shao, Y. Tian, Y. Zhang, et al., "Evolutionary computation for sparse multi-objective optimization: A survey," ACM Comput. Surv., in press.

[33] L. Si, X. Zhang, Y. Zhang, et al., "An efficient sampling approach to offspring generation for evolutionary large-scale constrained multi-objective optimization," IEEE Trans. Emerg. Top. Comput. Intell., in press.

[34] Y. Tian, X. Qi, S. Yang, et al., "A universal framework for automatically generating single- and multi-objective evolutionary algorithms," IEEE Trans. Evol. Comput., in press.